# Towards Online Mobile Mapping Using Inhomogeneous Lidar Data

Michiel Vlaminck[1], Hiep Quang Luong[1], Werner Goeman[2], Peter Veelaert[1] and Wilfried Philips[1]
Department of Telecommunications and Information Processing
[1]Ghent University, iMinds, Belgium
[2]Grontmij, Ghent, Belgium
Email: michiel.vlaminck@telin.ugent.be

***Abstract*— In this paper we present a novel approach to quickly obtain detailed 3D reconstructions of large scale environments. The method is based on the consecutive registration of 3D point clouds generated by modern lidar scanners such as the Velodyne HDL-32e or HDL-64e. The main contribution of this work is that the proposed system specifically deals with the problem of sparsity and inhomogeneity of the point clouds typically produced by these scanners. More specifically, we combine the simplicity of the traditional iterative closest point (ICP) algorithm with the analysis of the underlying surface of each point in a local neighbourhood. The algorithm was evaluated on our own collected dataset captured with accurate ground truth. The experiments demonstrate that the system is producing highly detailed 3D maps at the speed of 10 sensor frames per second.**

***Index Terms*— 3D point cloud registration, geometric primitive fitting, ICP, lidar scanning**

## I. Introduction

Mobile mapping has been widely studied over the past years. With the advent of modern lidar scanners such as the Velodyne HDL-16e, 32e or 64e, it has gained even more attention. These scanners are able to produce large point clouds, capturing the scene using respectively 16, 32 and 64 scan lines at approximately 10Hz. In the past, these systems have been successfully used on a mobile mapping van. However, with the growing presence of drones, researchers are getting inspired by mounting these lidar scanners on unmanned aerial vehicles (UAV's). As of March 2015, XactSense has been the first company to produce aerial mapping systems incorporating lidar technologies.

Until now, mobile mapping systems have mainly been used for mapping outdoor environments. As a result they are heavily relying on GPS to solve the *odometry problem*, i.e. the estimation of the vehicle's trajectory. However, one can easily think of environments where the GPS signal is too weak or even completely lacking. Therefore, a system that is fully GPS-independent and that is able to produce highly accurate 3D reconstructions of a scene is still sought after. A lot of solutions exist that rely on data originated from (omnidirectional) cameras to estimate the trajectory of the observer (cfr. visual odometry) and subsequently map the environment (cfr. photogrammetry). However, these solutions often result in sparse 3D maps and do not work well in bad weather conditions, at night or when the images are overexposed because of the sunlight. On the other hand, lidar sensors, although also slightly affected by bad weather conditions, are a lot more robust against these environmental conditions. In this paper we propose a novel approach to register consecutive 3D point clouds generated by spinning lidar scanners such as the Velodyne series. Our system is an online solution that incorporates loop closure in a very efficient way.

Some researchers have been experimenting with lidar data ([1], [2]), but most of their solutions are offline batch methods that are often computationally expensive and therefore unusable in certain applications. In addition, they all share the limitation of not dealing with the sparsity and inhomogeneity of the point clouds typically produced by lidar scanners. They often also assume that the sensor is put perpendicular to the ground plane which is not always feasible. Finally, some systems adopt a prior on the semantics of the scene, e.g. regarding the presence of large planar surfaces, which imposes an additional constraint on the system. The goal of this work is to create a robust and generic system that can perform 3D registration of point clouds in any scene without any assumption on the type of the scene or on the sensor set-up. Compared to other systems our solution does not assume any semantics of the environment, but in contrary, implicitly incorporates this information by analysing the underlying surface (i.e. being flat, spherical, cylindrical, ...). This way, the algorithm can benefit from the characteristics of the scene while still being generic and able to be applied in all kinds of scenarios.

## II. Related work

In the field of lidar odometry and mapping, the most cited algorithm for the registration of point clouds is the *iterative closest point (ICP)* algorithm mainly due to its simplicity. In [3] an extensive comparison was made between various ICP variants on real-world data sets. As the name suggests, ICP is an iterative algorithm that in each iteration tries to find corresponding (i.e. closest) points between the two point clouds based on their relative Euclidean distance. The algorithm subsequently estimates the rigid transformation between the two point clouds by minimizing the sum of all distances between the corresponding points. When the two point clouds are sufficiently close to each other, convergence is guaranteed. Many existing solutions are based on this ICP approach [4], [5], [6]. In the following, we will focus on those studies that use a spinning lidar scanner such as the Velodyne HDL. In [1], Moosmann *et al.* used the HDL-


Fig. 1. The mobile mapping van of Grontmij.

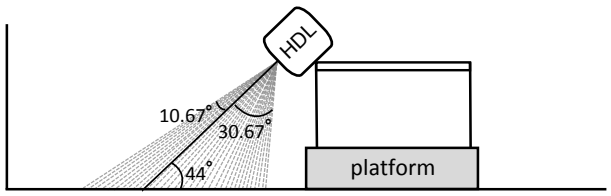


Fig. 2. Schematic drawing the mobile acquisition platform, consisting of a velodyne HDL-32e lidar scanner.

64e scanner in their work entitled *Velodyne SLAM*. The authors incrementally build a map of the environment by aligning each newly arrived scan with the whole map using the ICP algorithm. Since the scanner is spinning while it is moving, the point cloud associated with the scan is *skewed*. The authors therefore propose to perform *deskewing* as a preprocessing step. To this end, they assume that the velocity is constant during the acquisition of one scan. The most recent and prominent work regarding lidar SLAM is the research of Zhang et. al [2]. In their paper, the authors propose a real-time method using a 2-axis lidar moving in 6-DOF. Their system incorporates two processing loops, the first one being the estimation of the observer's trajectory (odometry), the other one being the actual mapping. The latter is running at a lower frequency to guarantee real-time performance. In [7], Grant *et al.* propose a novel algorithm to find large planes in (spinning) lidar data recorded in indoor scenes. The actual registration is solely based on the alignment of the set of planes as described in [8]. Finally, in [9], Ruiz *et al.* elaborate on the problem of fitting analytical surfaces in general to noisy point clouds. More specifically, they propose a method for automatically finding an initial guess for the analytic surface and describe how to economically estimate the geometric distance between a point and an analytical surface.

## III. System

### A. Acquisition platform

Our acquisition platform consists of a Velodyne High Definition lidar (HDL-32e) and is mounted on a mobile mapping van (see figure 1). As can be seen on figure 2, the sensor is tilted, making an angle of approximate 44 degrees with the ground plane. The Velodyne lidar scanner has 32 lasers covering a vertical FOV of 41.3° hence resulting in a vertical resolution of 1.29°. The vertical FOV covers 30.67 degrees below the middlepoint of the Velodyne and 10.67 degrees above it. The head is continuously spinning at approximately 10 Hz covering a horizontal FOV of 360 degrees. Although the Velodyne is on a moving vehicle we did not assume the fact that our platform is not able to rotate around its roll angle. A rotation around the pitch angle on the other hand is still possible in the case when the vehicle is riding uphill or downhill. The reason for this generalization is that we also want to be able to mount the Velodyne on a drone in the future. Our system thus operates as a full 6DoF SLAM algorithm incorporating three unknown position coordinates $x$, $y$, $z$ and three rotation angles $\theta_x$, $\theta_y$, $\theta_z$.

### B. Terminology

As mentioned in the previous section, the Velodyne is spinning its head, thereby producing 360° point clouds. A full rotation of the head is also referred to as a *sweep*. Throughout this paper, we use right subscript $k$, $k \in Z^+$ to indicate the sweep number and $\mathcal{P}_k$ to indicate the point cloud perceived during sweep $k$. The associated range image will be denoted by $\mathcal{R}_k$. Since we will project the point clouds onto a 2D grid (cfr. section IV-B), we will index it by image coordinates. We define the following functions.

$$\mathcal{P}_k(u,v) : (u,v) \mapsto (x,y,z)^\top$$
$$\mathcal{R}_k(u,v) : (u,v) \mapsto r$$

Alternatively, we will use right superscript $i$ to denote a single point $\mathbf{p}_k^i$ in the point cloud or a single range value $\mathrm{r}_k^i$ in the range image. The two data structures together will also be referred to as a *frame*. A frame is thus defined as follows.

$$\mathcal{F}_k = \{\mathcal{P}_k, \mathcal{R}_k\}$$

## IV. Approach

### A. Problem statement

The problem of mobile mapping usually involves solving the odometry problem, i.e. the estimation of the entire trajectory of the observer. If we know the relative poses of the sensor frames we can easily align them. However, since the platform is moving while the Velodyne's head is spinning, a problem arises when the motion of the observer is too high compared to the scan rate. In that case, the points acquired in one sweep are *skewed* because of latency due to the head rotation of the lidar scanner. In other words, points acquired at the beginning of a sweep are captured from a different location compared to points acquired at the end of the sweep. This poses an additional difficulty when we want to perform pairwise alignment of point clouds generated in two consecutive sweeps. A general approach therefore often consists of two processing loops, the first being the odometry, the second being the actual mapping. In this work, we adopt a slightly different approach. First we will *deskew* the points acquired during one sweep. Next, we perform pairwise alignment of consecutive, *deskewed*, point

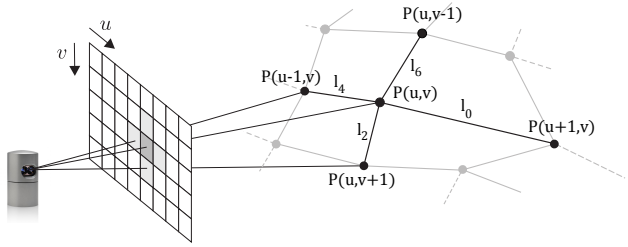


Fig. 3. Example of the 2D projection and adjacency property.

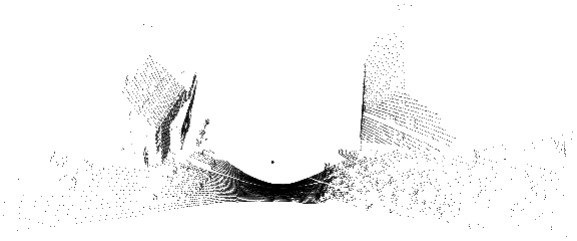

Fig. 4. An example of a point cloud $\mathcal{P}_k$ captured during sweep $k$ by the Velodyne HDL-32e at a street in the Belgian city Hasselt.

clouds. This results in a 3D map that is locally consistent, but since this pairwise alignment is an incremental process, it suffers from drift or error propagation. Therefore we will, in a final step, perform loop closure and subsequently pose graph optimization to make our 3D map globally consistent.

### B. 2D projection

The 3D points captured by the Velodyne are organized since the 32 lasers are placed collinear in the vertical direction. Because of that, we can project the 3D points onto a 2-dimensional spherical image in which each pixel represent a 3D point or a range value. Figure 5 shows an example of such a 360° range image. The blue color means that points are closeby whereas the red color denotes that points are located further away (recall that the sensor is tilted). Figure 3 gives a schematic representation of this projection. Using this 2D image we can exploit the adjacency in the pixel domain and adopt algorithms from the image processing literature to perform certain operations.

### C. Surface analysis

In order to perform surface analysis in the vicinity of a point $\mathbf{p}_k^i$, we need to determine the local neighbourhood of the point. This process is inherently time consuming, even when a k-dimensional tree is used. Fortunately, we can exploit the adjacency in the 2D domain. More precisely, we define the neighbourhood of a point $\mathbf{p}_k^i$ as the points lying at the distance of one *pixel* in the 2D image in each of the

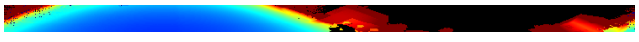

Fig. 5. A 360° spherical range image, obtained by projecting the point cloud $\mathcal{P}_k$ (see figure 4) onto a 2D grid. The blue color means that points are closeby whereas the red color denotes that points are located further away.

8 possible directions. Hence, we consider an 8-connected neighbourhood. Once the nearest neighbours are determined, we use them to estimate a local feature representation that captures the geometry of the underlying sampled surface. To this end, we perform principle component analysis (PCA) on each 8-connected surface patch. For each point $\mathbf{p}_k^i$ of $\mathcal{P}_k$ we construct the covariance matrix $C_k^i$ as follows.

$$C_k^i = \frac{1}{N}\sum_{j=1}^{N} w_j \cdot (\mathbf{p}_k^{\mathbf{i}(j)} - \bar{\mathbf{p}}_k^i) \cdot (\mathbf{p}_k^{\mathbf{i}(j)} - \bar{\mathbf{p}}_k^i)^T \quad (1)$$

$$C_k^i \cdot \mathbf{v}_l = \lambda_l \cdot \mathbf{v}_l, \quad l \in 0, 1, 2 \quad (2)$$

Herein, $N = 8$ is the number of neighbours of $\mathbf{p}_k^i$ that we take into account, $\lambda_l$ the $l$-th eigenvalue of the covariance matrix, $\mathbf{v}_l$ the $l$-th eigenvector and $\mathbf{i}$ a vector containing the indices of the neighbours of $\mathbf{p}_k^i$. The point $\bar{\mathbf{p}}_k^i$ is the 3-dimensional centroid of the nearest neighbours and is computed according to eq. 3.

$$\bar{\mathbf{p}}_k^i = \frac{1}{N} \cdot \sum_{j=1}^{N} \mathbf{p}_k^{\mathbf{i}(j)} \quad (3)$$

The eigenvector corresponding to the smallest eigenvalue represents the local surface normal of the point under consideration. Finally, $w_j$ is a weight factor that serves as a measure on how certain we are that point $\mathbf{p}_k^{\mathbf{i}(j)}$ is lying on the same surface as point $\mathbf{p}_k^i$. It can happen that $\mathbf{p}_k^{\mathbf{i}(j)}$ is located at the other side of a border between two surfaces and for this reason it should be excluded from the surface analysis. To determine these weight factors, we compute 8 differences in range values using the same 8 neighbouring points as before. For brevity, we will only formulate the definition of one value. The other values are defined in a similar way (see figure 3).

$$d(u,v)_0 = |\mathcal{R}(u+1,v) - \mathcal{R}(u,v)|$$

Finally, we define a measure $l$ on how *strong* the link between the 3D points is. Expression 4 gives the equation for the $l$-value of the link between $\mathcal{P}(u,v)$ and $\mathcal{P}(u+1,v)$. The other values are similar, but are again omitted.

$$l_0 = \min(\mathrm{f}(|\frac{d(u,v)_0 - d(u,v)_4}{d(u,v)_4}|), \mathrm{f}(|\frac{d(u,v)_0 - d(u+1,v)_0}{d(u+1,v)_0}|) \quad (4)$$

$$\mathrm{f}(x) = 1 - \mathrm{sigm}(x) \quad (5)$$

A high $l$-value indicates that the two points are likely lying on the same surface. A low $l$-value on the other hand indicates that the points are lying on both sides of a border between two surfaces. The reason why we consider the ratio between the range differences of two neighbouring points is because of the sparsity of points located further away. It can happen that there is a plane far away from the origin or with

a sharp inclination with respect to the sensor direction, which causes two neighbouring points to have a large difference in range value while still lying on the same plane. The weight $w_j$ is finally defined as in eq. 7.

$$\mathbf{l} = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7\} \tag{6}$$

$$w_j = \frac{l_j}{||\mathbf{l}||_1} \tag{7}$$

The sigmoid function serves as a soft threshold to penalize high ratio's of range values in order to exclude them from the estimation process. In the experiments we used the hyperbolic tangent function (tanh).

### D. Local registration

As mentioned in section II, the majority of existing systems rely on the ICP algorithm. Unfortunately, the standard ICP approach has some major limitations. One of the main problems has its origin in the fact that the point clouds generated by the Velodyne scanner typically have a sparse and inhomogeneous density. As a result, it is often impossible to find good point correspondences between two consecutive point clouds as most of the time the determined corresponding pairs will not represent the same physical point in space. This not only causes the algorithm to converge very slow but also results in point clouds that are not aligned accurately. For this reason we will not use the sampled points as target points, but use the feature point summarizing the local neighbourhood of each point. More precisely, we will use the local surface normal, computed as described in section IV-C. In the following we will briefly discuss the two main sub parts of the process.

*1) Correspondence estimation:* In contrast to existing systems we will select the corresponding *target* point $\mathbf{p}_{k-1}^j$ as the point having the most similar neighbourhood as the *source* point $\mathbf{p}_k^i$ in terms of the geometry of the underlying surface. To this end, we compute the distance from the point to the tangent plane of this corresponding surface. The target point in the vicinity of the source point that has the smallest distance is chosen as the corresponding point.

*2) Transformation estimation:* In traditional ICP approaches, the transformation between point clouds is estimated by minimizing the sum of the Euclidean distances between corresponding points after alignment. We, on the other hand, will not match points from the *source* point cloud $\mathcal{P}_k$ with points in the *target* point cloud $\mathcal{P}_{k-1}$, but rather match points of $\mathcal{P}_k$ with surface patches of $\mathcal{P}_{k-1}$. The goal is to minimize the distance between the points in $\mathcal{P}_k$ with the surface normal of the corresponding surface patch in $\mathcal{P}_{k-1}$. This results in the following error metric.

$$E(\mathcal{P}_k, \mathcal{P}_{k-1}; \hat{\mathrm{T}}_\mathrm{k}) = \sum_{i=1}^{N} ||(\hat{\mathrm{T}}_\mathrm{k}\mathbf{p}_k^i - \mathbf{p}_{k-1}^{\mathbf{c}(i)}) \cdot \mathbf{n}_{k-1}^{\mathbf{c}(i)}||^2 \tag{8}$$

Herein, $\hat{\mathrm{T}}_\mathrm{k}$ is the estimated transformation matrix, $\mathcal{P}_k$ is the *source* point cloud, $\mathcal{P}_{k-1}$ is the *target* point cloud, $\mathbf{n}_{k-1}^i$ is the surface normal according to target point $\mathbf{p}_{k-1}^i$ and $\mathbf{c}$ is the vector containing the indices of the $N$ corresponding points. Eq. 9 gives the expression to derive the final transformation matrix $\mathrm{T}_k$.

$$\hat{\mathrm{T}}_\mathrm{k} = \begin{bmatrix} \hat{\mathrm{R}}_\mathrm{k} & \hat{\mathbf{t}}_\mathbf{k} \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \underset{\hat{\mathrm{T}}_\mathrm{k}}{\operatorname{argmin}}\, E(\mathcal{P}_k, \mathcal{P}_{k-1}; \hat{\mathrm{T}}_\mathrm{k}) \tag{9}$$

In order to solve this optimization problem, we adopt the method proposed by Low *et al.* in [10]. In that paper, a method is derived to approximate the nonlinear optimization problem with a linear least squares one in case the relative orientation between the two input point clouds is small. Since we consider point clouds acquired in two consecutive sweeps, this assumption is guaranteed in our case.

### E. Global pose graph optimization

So far we have only been discussing the *local* registration of point clouds acquired during two consecutive sweeps. As this is an incremental process, small errors can propagate causing a considerable amount of drift. To cope with this drift, we will perform loop closure and subsequently pose graph optimization. At this moment we have a graph in which each node consists of a pose, i.e. position and orientation in 6D, of a sensor frame. Since we want to obtain the 3D model in real time, it is important that this loop closure only takes a few milliseconds. Therefore we use the explicit loop closing heuristic (ELCH) algorithm presented in [11].

## V. Evaluation

To evaluate our system we recorded a data sequence in the streets of the Belgian city Hasselt. The exact longitude and latitude of the starting point are respectively $5.30935°$ and $50.9416°$. An image of this starting point is depicted in figure 6. The mobile mapping van was driving at a speed of approximately 15 km/h. Besides data captured with the Velodyne scanner, we also recorded accurate positioning information using the POS LV 420 positioning sensor developed by Applanix. This INS system incorporates both an inertial measurement unit (IMU), a distance measurement indicator (DMI) and Trimble's BD960 GNSS receiver. Since each point of the Velodyne data is timestamped we can derive the exact geolocation of each point in space. We used the Belgian Lambert72 geographic information system (GIS), based on the ellipsoid of Hayford, to express this geolocation. To be able to evaluate our outcome against the ground truth we transformed both the ground truth trajectory $\mathbf{T} = \{\mathrm{T}_1, \ldots, T_k, \ldots, \mathrm{T}_N\}$ and estimated trajectory $\hat{\mathbf{T}}' = \{\hat{\mathrm{T}}'_1, \ldots, \hat{\mathrm{T}}'_k, \ldots, \hat{\mathrm{T}}'_N\}$ into the same reference system. We decided to set the origin of the first laser frame as global origin. As such we define an initial transformation $T_0$ given by eq. 10.

$$\mathrm{T}_0 = \begin{bmatrix} \mathrm{R}_0 & \mathbf{t}_0 \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \mathrm{T}_0^{(\mathrm{las2geo})} = \mathrm{T}_0^{(\mathrm{veh2geo})}\mathrm{T}^{(\mathrm{las2veh})} \tag{10}$$

The matrix $\mathrm{T}^{(\mathrm{las2veh})}$ represents the transformation from the laser frame to the vehicle frame and is obtained by

Fig. 6. Image of the starting point obtained by google street view.

accurate calibration. The vector of transformation matrices $\mathbf{T}^{(\text{veh2geo})} = \{T_0^{(\text{veh2geo})}, \ldots, T_k^{(\text{veh2geo})}, \ldots, T_N^{(\text{veh2geo})}\}$ is derived from the POS LV positioning system at the timestamp of the first point $\mathbf{p}_k^0 \in \mathcal{P}_k$ acquired during sweep $k$. The first pose of the ground truth trajectory is given by eq. 11.

$$\mathrm{T}_1 = \mathrm{T}_1^{(\text{veh2geo})}\mathrm{T}_0^{-1} \tag{11}$$

The rest of the ground truth trajectory is given by eq. 12.

$$\mathrm{T}_\mathrm{k} = \begin{bmatrix} \mathrm{R}_\mathrm{k} & \mathbf{t}_k \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \mathrm{T}_\mathrm{k}^{(\text{veh2geo})}\mathrm{T}_{\mathrm{k}-1} \tag{12}$$

Similarly, for the estimated trajectory, we will put each sensor frame in the same reference system by transforming it using the same rotation $\mathrm{R}_0$. We thus define the initial transformation matrix $\hat{\mathrm{T}}'_0$ as follows (eq. 13).

$$\hat{\mathrm{T}}'_0 = \begin{bmatrix} \mathrm{R}_0 & \mathbf{0}_3 \\ \mathbf{0}_3^\top & 1 \end{bmatrix} \tag{13}$$

The estimated trajectory is given by eq. 14.

$$\hat{\mathrm{T}}'_k = \begin{bmatrix} \hat{\mathrm{R}}'_\mathrm{k} & \hat{\mathbf{t}}'_k \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \hat{\mathrm{T}}_\mathrm{k}\hat{\mathrm{T}}'_{k-1} \tag{14}$$

The matrix $\hat{\mathrm{T}}_\mathrm{k}$ is the result of our registration defined in eq. 9. The reconstructed 3D point cloud $\mathcal{P}_{world}$ of the entire sequence is finally given by eq. 15. A visualization of the result is depicted in figure 7.

$$\mathcal{P}_{world} = \{\mathcal{P}_0, \ldots, \hat{\mathrm{T}}'_k\mathcal{P}_k, \ldots, \hat{\mathrm{T}}'_N\mathcal{P}_N\} \tag{15}$$

Yet, in order to be able to evaluate our system quantitatively, we projected the exact and estimated position of the vehicle onto the ground plane and derived the translational error $e_t$ and rotational error $e_r$ from it. Since we have set the first sensor frame as the origin, the exact and estimated position of the vehicle at the start of sweep $k$ is given by respectively $\mathbf{t}_k$ and $\hat{\mathbf{t}}'_k$. The projection on the ground plane is given by function 16. Recall that our original sensor was tilted by $44°$. However, the rotation $\mathrm{R}_0$ aligned the ground plane with the $xz$-plane in the final reference system.

$$G(x, y, z) : (x, y, z) \mapsto (x, z) \tag{16}$$

A plot of the ground truth and estimated trajectory on top of the corresponding google earth image is depicted in figure 8. We also included the outcome of the standard point-to-point ICP algorithm after loop closure. As can be seen, the transformation estimation is very poor due to the shortcomings described in section IV-D. In order to evaluate the rotational and translational error we quantise the trajectory in intervals of length $\Delta$. The sensor pose $\mathrm{T}_f$ corresponding with frame $f$ represents the first sensor pose of the interval, whereas $\mathrm{T}_l$ represents the last sensor pose of the interval. For $\mathrm{T}_f$ and $\mathrm{T}_l$ the following condition is valid (eq. 17).

Fig. 7. The resulting point cloud $\mathcal{P}_{world}$ of our mobile mapping system on the entire lidar sequence recorded in the Belgian city of Hasselt.

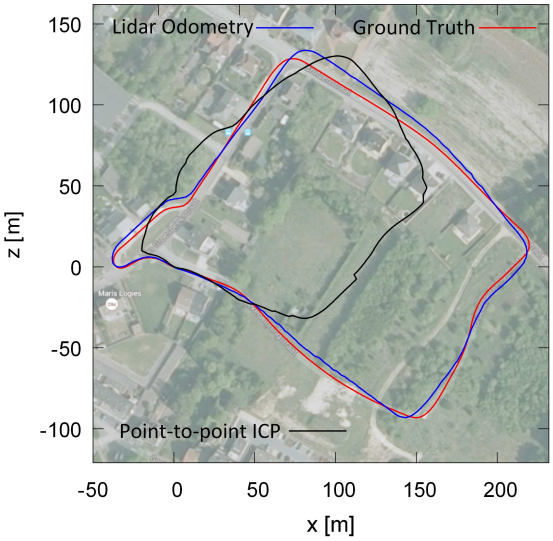


Fig. 8. A plot of the ground truth trajectory $\mathbf{T}$ (red) as well as the estimated trajectory $\hat{\mathbf{T}}'$ (blue) on top of the corresponding google earth image. By way of comparison, the trajectory obtained by performing standard point-to-point ICP is plotted in black.

$$||\mathbf{t}_f - \mathbf{t}_l||_2 \approx \Delta \tag{17}$$

We now define the *difference* of the two ground truth poses $\mathrm{T}_f$ and $\mathrm{T}_l$ and estimated poses $\hat{\mathrm{T}}'_f$ and $\hat{\mathrm{T}}'_l$ as follows (eq. 18 and eq. 19).

$$T_\Delta = T_f^{-1}T_l \tag{18}$$

$$\hat{T}_\Delta = \hat{T}'^{-1}_f\hat{T}'_l \tag{19}$$

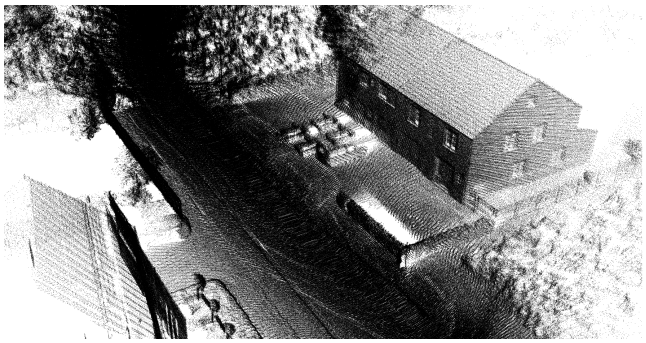

Fig. 9. Zoom in on a part of the reconstructed point cloud.

The final *pose error* $\mathrm{E}_\Delta$ is than given by eq. 20.

$$\mathrm{E}_\Delta = \begin{bmatrix} \mathrm{R}_\mathrm{e} & \mathbf{t}_e \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \hat{T}_\Delta^{-1} T_\Delta \tag{20}$$

Finally, the rotational error $e_r$ and translational error $e_t$ is given by eq. 22 and eq. 23 respectively.

$$\mathrm{d} = \frac{1}{2}(\mathrm{tr}(\mathrm{R}_e) - 1) \tag{21}$$

$$e_r = \frac{1}{\Delta}\mathrm{acos}(\max(\min(d, 1), -1)) \tag{22}$$

$$e_t = \frac{1}{\Delta}||\mathbf{t}_e||_2 \tag{23}$$

A plot of $e_r$ and $e_t$ for $\Delta \in \{100, \ldots, 700\}$ is depicted in figure 10. Regarding the rotation accuracy we see that $e_r$ lies within the interval of 0.16 to 0.01 degrees per meter whereas $e_t$ ranges from 0.2 to below 0.01. The rather high values for small intervals are particularly caused by areas that lack geometrical properties due to the vegetation along the way instead of buildings. Further, we see that for increasing intervals, the errors are greatly reduced. This is mainly due to the loop closure and pose graph optimization. Finally, we can report a processing speed of approximately 10Hz including the loop closure, which outperforms most of the existing solutions.

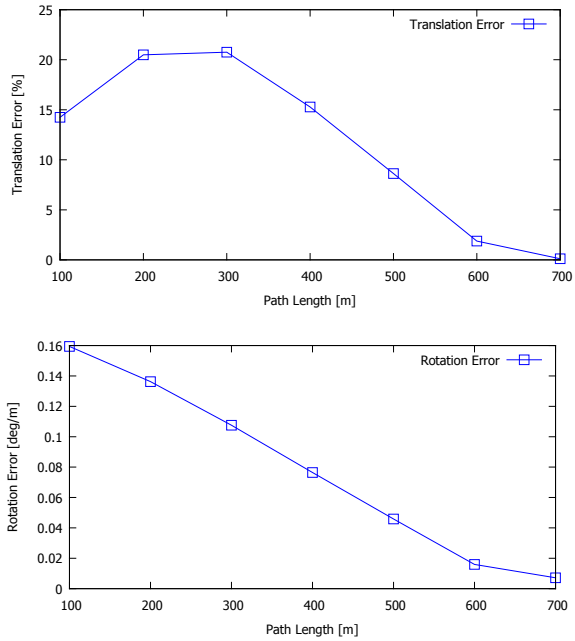


Fig. 10. Translational and rotational error on our data sequence.

## VI. Conclusion

In this paper, a novel mobile mapping system using lidar data was presented. The solution is unique in the sense that the standard ICP approach is modified to cope with the sparsity and inhomogeneity of the point clouds produced by lidar scanners. To this end, an additional surface analysis is performed in the local neighbourhood of each point. The system is running at approximately 10 frames per second, which is faster than most existing solutions.

## Acknowledgement

This research is part of the GiPA project, an ICON project co-funded by iMinds, digital research institute founded by the Flemish Government. The project partner is Grontmij, with project support from IWT.

## References

[1] F. Moosmann and C. Stiller, "Velodyne SLAM," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 393–398. [Online]. Available: http://www.mrt.kit.edu/z/publ/download/Moosmann_IV11.pdf

[2] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems Conference (RSS)*, Berkeley, CA, July 2014.

[3] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Auton. Robots*, vol. 34, no. 3, pp. 133–148, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-013-9327-2

[4] R. Zlot and M. Bosse, "Efficient large-scale 3d mobile mapping and surface reconstruction of an underground mine." in *FSR*, ser. Springer Tracts in Advanced Robotics, K. Yoshida and S. Tadokoro, Eds., vol. 92. Springer, 2012, pp. 479–493. [Online]. Available: http://dblp.uni-trier.de/db/conf/fsr/fsr2012.html#ZlotB12

[5] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[6] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam - 3d mapping outdoor environments: Research articles," *J. Field Robot.*, vol. 24, no. 8-9, pp. 699–722, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1002/rob.v24:8/9

[7] W. Grant, R. Voorhies, and L. Itti, "Finding planes in lidar point clouds for real-time registration," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, bb, pp. 4347–4354.

[8] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3d mapping," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

[9] O. Ruiz, S. Arroyave, and D. Acosta, "Fitting of analytic surfaces to noisy point clouds," *American Journal of Computational Mathematics*, vol. 3, no. 1A, pp. 18–26, 2013, special issue on Computational Geometry. doi:10.4236/ajcm.2013.31A004.

[10] K.-L. Low, "Linear least-squares optimization for point-to plane icp surface registration," Tech. Rep., 2004.

[11] J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg, "An explicit loop closing technique for 6d slam." in *ECMR*, I. Petrovic and A. J. Lilienthal, Eds. KoREMA, 2009, pp. 229–234.